

## 3.2) 幂等性以及消息的幂等性

### 一:什么是接口的幂等性?

**接口的幂等性:**简而言之,就是对接口发起的一次调用和多次调用,锁生产的结果都是一致的。

**某些接口具有天然的幂等性:**比如长查询接口,不管是查询一次还是多次,返回的结果都是一致的

### 1.2) 若接口没有保障幂等性,那么就出现问题

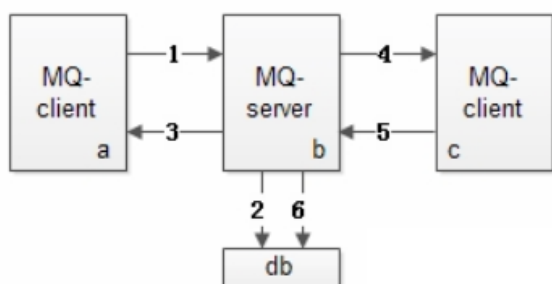
**案例一:**比如订单提交的过程中,用户点了一次提交,但是由于网络等原因,导致后端处理延时,客户就连续点击了多次,在没有幂等性的条件下,那么就会造成订单的重复提交。

**解决方案:**在保存订单的时候,根据生成的系统全局唯一ID(这里订单号+业务类型),并且把该唯一ID 调用 redis 的setnx命令保存起来,在第一次保存的时候,由于redis中没有该key,那么就会

把全局唯一ID 进行设置上,此时订单就会保存成功,。这个时候若出现前端重复点击按钮,由于第一步已经setnx上了 就会阻止后面的保存。

### 二: MQ 是如何解决幂等性的

#### 发送消息的流程



**第一步:**消息生产者向Mq服务端发送消息

**第二步:**mq 服务端把消息进行落地

**第三步:**消息服务端向消息生产者发送ack

**第四步:**消息消费者消费消息

**第五步:**消费者发送ack

## 第六步: mq服务将落地消息删除

### 2.1) 消息重复发送的原因

为了保障消息的百分之百的投递，我们使用了消息重发，确认机制，使得消息可能被重复发送，由上图可知，由于网络原因，第三步的上半场ack丢失还是第五步的下半场ack丢失

都会导致消息重复发送

### 2.2) 消息重复发送的导致后果.

上半场消息生产者为用户支付模块，专门是用来给用户扣费的，而下半场的消息消费者服务是会员卡服务，是通过接受扣费服务发送的消息来进行发卡的，

由于第三步或者是第五步ack丢失，那么就会导致上游服务重复发送消息就会导致扣一次款，发多次卡

### 2.3)mq服务端是如何保证幂等性的？

消息队列的服务中，对每一条消息都会生成一个全局唯一的与业务无关的ID(inner\_msg\_id),当mq\_server接受到消息的时候，先根据inner\_msg\_id 是否需要重复发送，再决定消息是否落DB ,这样保证每条消息都只会落一次DB

### 2.4) 消费端如何来做到幂等性的？

还是把对每条消息做生成一个唯一性的ID 通过redis的来setnx命令来保证幂等性。